REMARKS

The Office action has been carefully considered. The Office action rejected claims 1-35 under 35 U.S.C. § 102(a) as being anticipated by *Managing PC Operating Systems with a Revision Control System*, published in October 1997 to Gottfried Rudorfer, ("Rudorfer"). Further, the Office action rejected claims 1, 12, 21, 26, 30, and 35 under the judicially-created doctrine of obviousness-type double patenting as having recitations that are obvious variations of the claims of U.S. Patent No, 6,427,277 B1 assigned to the same assignee of the present application. Applicants submit herewith a terminal disclaimer to overcome this double-patenting rejection. Regarding the §102 rejection, applicants respectfully disagree.

By present amendment, claims 1, 12 and 30 have been amended for clarification and not in view of the prior art. Applicants submit that the claims as filed were patentable over the prior art of record, and that the amendments herein are for purposes of clarifying the claims and/or for expediting allowance of the claims and not for reasons related to patentability. Reconsideration is respectfully requested.

Applicants thank the Examiner for the interview held (by telephone) on November 16, 2005. During the interview, the Examiner and applicants' attorney discussed the claims with respect to the prior art. The essence of applicants' position is incorporated in the remarks below.

Prior to discussing reasons why applicants believe that the claims in this application are clearly allowable in view of the teachings of the cited and applied references, a brief description of the present invention is presented.

The present invention is directed to a system and method for providing a computer system the ability to repair an installed application if a needed resource is identified as not available due to inadvertent removal, transfer, or deletion. As can be appreciated, applications draw from many different files when executing, such as direct link library files, data files, content files, *etc*. Sometimes, these files become corrupted, removed, or otherwise unavailable during the operation of the application and the application becomes lost or stalled. In the present invention, however, the application may instead be modified to actively locate the misplaced or corrupted file in an effort to repair itself without user intervention. As such, the application need not terminate just because a file that is needed cannot be located.

In one example embodiment, a computer system may include already installed code therein, including an executable software program that has a first set of executable code and a second set of executable code. Typically, the first set of executable code may be the crux of an application, *e.g.*, a word processing program (such as in an .exe file), and the second set of executable code may be a resource that is needed to provide functionality to the first set of executable code, *e.g.*, a search-related file (such as including a .dll file). The system may further include an installer program configured to determine necessary installation information for specific files (like the .dll file), and when the installation information indicates that the set of executable code is not installed, the installer program may be further configured to automatically install the second set of executable code to make the second set of executable code available to the first set of executable code, e.g., while the main application itself remains installed and executing. Thus, the application need not shut down or reinstall simply because some file is missing or corrupt.

Note that the above description is for example and informational purposes only, and should not be used to interpret the claims, which are discussed below.

Turning to the claims, independent claim 1 generally recites receiving a request to verify that a needed resource is available to an executable software program that is already installed, the needed resource comprising at least one resource needed by the program and determining whether the needed resource is available to the program, and if the needed resource is not available to the program, automatically initiating an installation procedure without manual termination of the program to make the needed resource available to the program.

The Office action rejected claim 1 as being anticipated by Rudorfer. More specifically, the Office action contends that Rudorfer teaches receiving a request to verify that a needed resource is available to an executable software program, the needed resource comprising at least one resource needed by the program and determining whether the needed resource is available to the program. Pages 79-80 and steps 1-8 of Rudorfer are referenced. The Office action contends further that Rudorfer teaches that if the needed resource is not available to the program, automatically initiating an installation procedure without manual termination of the program to make the needed resource available to the program. Pages 80, 82, and steps 1-8 of Rudorfer are again referenced. Applicants respectfully disagree.

Rudorfer is directed to a system for managing the deployment of applications in a networked computing environment. In specific, Rudorfer describes a server-client relationship wherein a system administrator may create a master copy of a typical application set on a server computer for deployment to all connected client machines. In this manner, a user at a client machine need only step through well-defined procedures for updating and/or installing applications at each respective client machine. To this end, Rudorfer describes (e.g., via steps 1-8 on page 80 of Rudorfer) a Perl script that may be executed to accomplish the various aspects of Rudorfer's method. These steps, however, are rudimentary steps that a client machine may undertake when being reinitialized, formatted, deployed or otherwise initiated for the first time ever. Thus, the system in Rudorfer assumes that the client machine (or at least the volume about to be formatted) is a fresh environment void of any installed applications at all. Despite not having to reboot after initialization, Rudorfer remains an example of prior art that teaches completely installing applications and the like at the request of a server computer or a script initiated therefrom.

In contrast, claim 1 recites receiving a request to verify that a needed resource is available to an executable software program that is already installed. Quite different from Rudorfer, claim 1 is directed to a system that deals with requests for resources from applications that are already installed. That is, when an application is already installed and executing, needed resources that are not found in the location as expected may be requested (via a request to an installer component) by the application. Rudorfer is wholly unaware of any procedure for dealing with requests from applications that are already installed, since the heart of Rudorfer's teachings necessarily involves installing these very applications in the first place. As such, Rudorfer cannot be construed to teach receiving a request to verify that a needed resource is available to an executable software program that is already installed as recited in claim 1.

Furthermore, claim 1 recites automatically initiating an installation procedure without manual termination of the program to make the needed resource available to the program. That is, if the needed resource is not available when requested by the application that is already installed and even, perhaps, running, then an installation procedure is automatically initiated in an effort to locate and install the needed resource all while the application remains installed and executing. Rudorfer cannot possibly do this as all of Rudorfer's applications in questions are deployed via the installation procedure. Rudorfer describes a process for streamlining the deployment of application sets to client computers via a server. Additionally, the method described by Rudorfer still requires user intervention in initiating the installation process, despite the installation process being embodied in a Perl script that executes automatically after initialization. Requiring user intervention to begin the process of installing applications is not the same as automatically initiating an installation procedure without manual termination of the program to make the needed resource available to the program as recited in claim 1. Applicants submit that Rudorfer does not teach all of the recitations of claim 1 and that claim 1 is allowable over the prior art of record for at least the foregoing reasons.

Applicants respectfully submit that dependent claims 2-11, by similar analysis, are allowable. Each of these claims depends either directly or indirectly from claim 1 and consequently includes the recitations of independent claim 1. As discussed above, Rudorfer fails to disclose the recitations of claim 1 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 1 noted above, each of these dependent claims includes additional patentable elements.

For example, claim 3 essentially recites that the resource may comprise a file, and that automatically initiating an installation procedure further comprises installing the file at a storage location accessible to the executable software program. The method of Rudorfer describes installing a whole application, which is not, in fact, a single file in and of itself. For at least this additional reason, applicants submit that claim 3 is allowable over the prior art of record.

Turning to the next independent claim, amended claim 12 generally recites an executable software program including a first set of executable code that is already installed and a second set of executable code, the second set of executable code comprising at least one resource that is needed to provide functionality to the first set of executable code and an installer program connected for communication with the first executable software program to receive a request for installation information of the second set of executable code, the installer program configured to determine the installation information, and when the installation information indicates that the set of executable code is not installed, the installer program further configured to automatically install the second set of executable code to make the second set of executable code available to the first set of executable code.

The Office action rejected claim 12 as being anticipated by Rudorfer. More specifically, the Office action contends that Rudorfer teaches an executable software program including a first set of executable code and a second set of executable code, the second set of executable code comprising at least one resource that is needed to provide functionality to the first set of executable code. Page 79 and 80 of Rudorfer are referenced. Further, the Office action contends that Rudorfer teaches an installer program connected for communication with the first executable software program to receive a request for installation information of the second set of executable code, the installer program configured to determine the installation information, and when the installation information indicates that the set of executable code is not installed, the installer program further configured to automatically install the second set of executable code to make the second set of executable code available to the first set of executable code. Again, pages 79 and 80 of Rudorfer are referenced. Applicants respectfully disagree.

As discussed above, Rudorfer teaches a system and method for installing a set of applications according to a predefined deployment scheme for a large number of client machines coupled to a server. As has been shown above, all installed applications of Rudorfer are done as an initial installation with no applications that are already present at the client machine being installed. In fact, the whole point of the system and method of Rudorfer is to deploy specific kinds of applications for use on several client machines in a grouping of client machines coupled to the server.

In contrast to Rudorfer, claim 12 recites an installer program connected for communication with the first executable software program to receive a request for installation information of the second set of executable code. In a different manner than Rudorfer, claim 1 is directed to a system that deals with requests for resources from applications that are already installed. That is, when an application is already installed and executing, needed resources, that are not found in the location as expected may be requested (via a request to an installer component) by the application. Rudorfer is wholly unaware of any procedure for dealing with requests from applications that are already installed, since the point of Rudorfer's method necessarily involves installing these very applications in the first place. As such, Rudorfer cannot be reasonably construed to teach an installer program connected for communication with the first executable software program to receive a request for installation information of the second set of executable code as recited in claim 12.

Additionally, claim 12 recites the installer program further configured to automatically install the second set of executable code to make the second set of executable code available to the first set of executable code. That is, if the needed resource is not available when requested by the application that is already installed and even, perhaps, running, then an installation procedure is automatically initiated in an effort to locate and install the needed resource all while the application remains installed and executing. Rudorfer cannot possibly do this as all applications in question are deployed via the installation procedure as Rudorfer describes a process for streamlining the deployment of application sets to client computers via a server. Additionally, the model described by Rudorfer still requires user intervention in initiating the installation process despite the installation process being embodied in a Perl script that executes automatically after initialization. Requiring user intervention to begin the process of installing applications is not the same as automatically installing the second set of executable code to make the second set of executable code available to the first set of executable code as recited in claim 12. Applicants submit that Rudorfer does not teach all of the recitations of claim 12 and that claim 12 is allowable over the prior art of record for at least the foregoing reasons.

Applicants respectfully submit that dependent claims 13-20, by similar analysis, are allowable. Each of these claims depends either directly or indirectly from claim 12 and consequently includes the recited limitations of independent claim 12. As discussed above, Rudorfer fails to disclose the recited limitations of claim 12 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 12 noted above, each of these dependent claims includes additional patentable elements.

Turning to the next claim, amended claim 21 generally recites receiving a resource identifier comprising at least one argument from a first set of executable code that is already installed, the resource identifier being associated with a second set of executable code including at least one resource that provides functionality to the first set of executable code, accessing a database based on the resource identifier to retrieve an expected location of at least part of the second set of executable code, and verifying the existence of the at least part of the second set of executable code at the expected location, and if verification is positive, passing the expected location to the first set of executable code.

The Office action rejected claim 21 as being anticipated by Rudorfer. More specifically, the Office action cites the same sections of Rudorfer and puts forth the same rationale as previously presented with respect to the rejection of claim 1. Applicants respectfully disagree.

Similar to the reasons put forth above, claim 21 recites language that is unmistakably not taught by Rudorfer. Specifically, claim 21 recites receiving a resource identifier comprising at least one argument from a first set of executable code that is already installed, the resource identifier being associated with a second set of executable code including at least one resource that provides functionality to the first set of executable code. Again, the system of Rudorfer does not teach receiving any request for installation of needed resources from anything that is already installed.

Furthermore, claim 21 recites accessing a database based on the resource identifier to retrieve an expected location of at least part of the second set of executable code. Claim 21 goes on to recite verifying the existence of the at least part of the second set of executable code at the expected location, and if verification is positive, passing the expected location to the first set of executable code. These are concepts that cannot possibly be taught by Rudorfer as the nature of the verification process begins and ends with an already installed application. Rudorfer necessarily teaches dealing with application installations for applications not installed on the client machine. Applicants submit that claim 21 is allowable over the prior art of record for at least the foregoing reasons.

Applicants respectfully submit that dependent claims 22-25, by similar analysis, are allowable. Each of these claims depends either directly or indirectly from claim 21 and consequently includes the recitations of independent claim 21. As discussed above, Rudorfer fails to disclose the recitations of claim 21 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 21 noted above, each of these dependent claims includes additional patentable elements.

Turning to the next independent claim, claim 26 generally recites executable code having a feature, the feature comprising a component including a key file which supports the feature and an installer for repairing the executable code if the key file becomes unavailable to the executable code, the installer (a) receiving from the executable code a request for a path to the key file, (b) identifying an expected location for the key file in the computing environment, (c) attempting to verify the existence of the key file at the expected location in the computing environment and (d) in response to a failure to verify the existence of the key file at the expected location, automatically initiating an installation of the key file to the expected location without manual termination of the executable code.

The Office action rejected claim 26 as being anticipated by Rudorfer. More specifically, the Office action cites the same sections of Rudorfer and puts forth the same rationale as previously presented with respect to the rejection of claim 1. Applicants respectfully disagree.

Applicants submit that claim 26 is allowable over the prior art of record for similar reasons as have been discussed above with respect to all previous claims. In specific, claim 26 recites executable code having a feature, the feature comprising a component including a key file which supports the feature. Rudorfer does not teach any such application. Further, claim 26 recites, in response to a failure to verify the existence of the key file at the expected location, automatically initiating an installation of the key file to the expected location without manual termination of the executable code. Again, Rudorfer does not teach installing anything in response to a failure to locate specific code and certainly not in an automatic manner. For at least the foregoing reasons, applicants submit that claim 26 is allowable over the prior art of record.

Applicants respectfully submit that dependent claims 27-29, by similar analysis, are allowable. Each of these claims depends either directly or indirectly from claim 26 and consequently includes the recitations of independent claim 26. As discussed above, Rudorfer fails to disclose the recitations of claim 26 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 26 noted above, each of these dependent claims includes additional patentable elements.

Turning to the last independent claim, amended claim 30 generally recites receiving a call from an installed application, the call including a resource identifier and in response to receiving the call: 1) determining if a resource corresponding to the resource identifier exists at an expected location, and if the resource does not exist at the expected location, automatically initiating an installation of the resource to the expected location; and 2) returning information corresponding to the existence of the resource at the expected location.

The Office action rejected claim 30 as being anticipated by Rudorfer. More specifically, the Office action cites the same sections of Rudorfer and puts forth the same rationale as previously presented with respect to the rejection of claim 1. Applicants respectfully disagree.

Once again, applicants submit that claim 30 is allowable over the prior art of record for similar reasons as have been discussed above with respect to all previous claims. In specific, claim 30 recites receiving a call from an installed application, the call including a resource identifier. Rudorfer does not teach any application that is currently installed. Further, claim 30 recites if the resource does not exist at the expected location, automatically initiating an installation of the resource to the expected location. Again, Rudorfer does not teach installing anything in response to a failure to locate specific code and any automatic installation of applications is only done so at the request of a client user or a system administrator. For at least the foregoing reasons, applicants submit that claim 30 is allowable over the prior art of record.

Applicants respectfully submit that dependent claims 31-35, by similar analysis, are allowable. Each of these claims depends either directly or indirectly from claim 30 and consequently includes the recitations of independent claim 30. As discussed above, Rudorfer fails to disclose the recitations of claim 30 and therefore these claims are also allowable over the prior art of record. In addition to the recitations of claim 30 noted above, each of these dependent claims includes additional patentable elements.

For at least these additional reasons, applicants submit that all the claims are patentable over the prior art of record. Reconsideration and withdrawal of the rejections in the Office action is respectfully requested and early allowance of this application is earnestly solicited.

CONCLUSION

In view of the foregoing remarks, it is respectfully submitted that claims 1-35 are patentable over the prior art of record, and that the application is in good and proper form for allowance. A favorable action on the part of the Examiner is earnestly solicited.

If in the opinion of the Examiner a telephone conference would expedite the prosecution of the subject application, the Examiner is invited to call the undersigned attorney at (425) 836-3030.

Respectfully submitted,

Albert S. Michalik, Reg. No. 37,395
Attorney for Applicants
Law Offices of Albert S. Michalik, PLLC
704 - 228th Avenue NE, Suite 193
Sammamish, WA 98074
(425) 836-3030
(425) 836-8957 (facsimile)